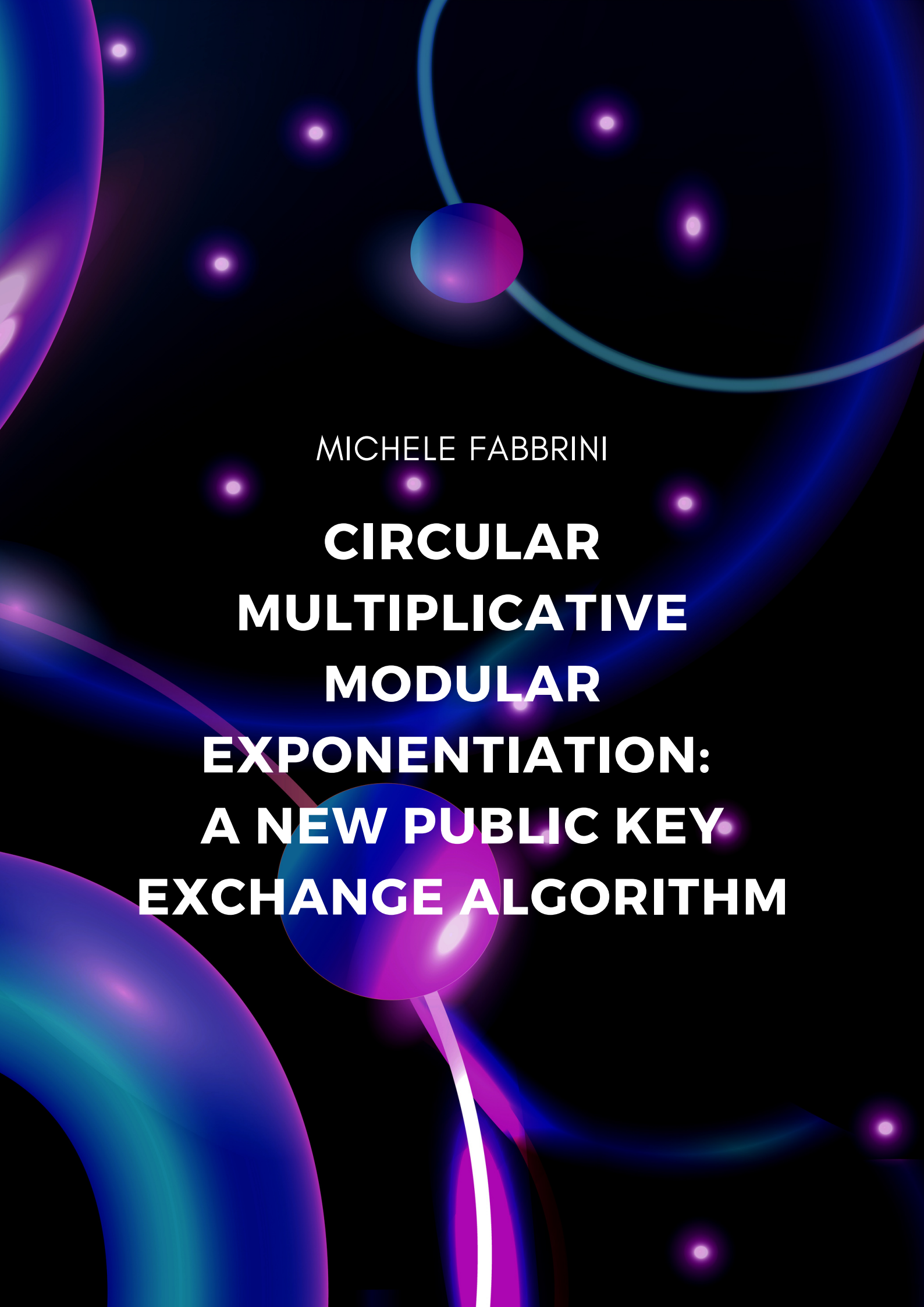


MICHELE FABBRINI

CIRCULAR MULTIPLICATIVE MODULAR EXPONENTIATION: A NEW PUBLIC KEY EXCHANGE ALGORITHM

Circular Multiplicative Modular Exponentiation: A New Public Key Exchange Algorithm

Author: **Michele Fabbrini**
First published: **June 8, 2023**
Doi: **10.13140/RG.2.2.25363.45602**
License: **Attribution 4.0 International (CC BY 4.0)**

Circular Multiplicative Modular Exponentiation: A New Public Key Exchange Algorithm

Michele Fabbrini*

June 19, 2023

Abstract

The major objective of this paper is to present a theoretical model for an algorithm that has not been previously described in the literature, capable of generating a secret key through the transmission of data over a public channel. We explain how the method creates a shared secret key by attaining commutativity through the multiplication of the modular exponentiation of a minimum of two bases and an equal number of private exponents for each party involved in the exchange. In addition, we explore the relationship between CMME and the traditional Diffie-Hellman scheme. We just briefly discuss the algorithm's security, opting to leave the essential investigation to cryptanalysts, while we elucidate on the algorithm's mechanism by illustrating some cases.

Keywords— public key cryptography, multiplicative modular exponentiation, public key exchange scheme, key distribution problem

1 Introduction

The Key Distribution Problem (KDP) is the challenge of effectively transmitting keys to users who require them. Ralph C. Merkle is credited with publishing the first scientific paper on Public Key Cryptography in the fall of 1974 [1]. Prior to the 1970s, anyone who wanted to use encryption had to deal with this issue until Whitfield Diffie and Martin Hellman [2], and GCHQ independently discovered a way for distributing keys without transferring the keys themselves. Some years ago GCHQ acknowledged that Clifford Cocks, one of their employees, had developed basically the same idea four years earlier, motivated by James Ellis's publications on the feasibility of cryptography without a secret key [3] (the discovery however was kept secret for over 45 years [4]). Since then, public key cryptography has evolved, seeking new paths, but the fascination for the linearity and extraordinary effectiveness of that first algorithm remains unchanged and continues to arouse the wonder of generations of cryptographers. A one-way function with a trapdoor and some type of mathematical process that ensures commutativity [5] are required to construct a good public key exchange algorithm. A one-way function is a function for which computation in one direction is simple, but computation in the opposite direction is extremely challenging. A trapdoor, on the other hand, is a one-way function with a further feature: it can be efficiently reversed if the secret information "trapdoor" is known. As far as commutativity is concerned, in the Diffie-Hellman scheme for example we find it in the exponentiation: $(g^a)^b = (g^b)^a$. The basic Diffie–Hellman scheme implements the multiplicative group of integers modulo a prime number p. Since the discrete logarithm problem (DLP) on this group can be solved in subexponential time in the standard model [6] by substituting a different cyclic group, alternative versions of the original scheme were pursued. In this regard, see the remarkable research by Koblitz [7] and Miller [8] on elliptic curves. Neverthless, using Shor's algorithm, all discrete logarithm-based schemes can be broken in polynomial time in the quantum computation model [10]. The supersingular isogeny Diffie–Hellman key exchange (SIDH) scheme represents an attempt to go beyond the usual approach and is currently being intensively investigated [9]. Our research shares with the latter the desire to transcend the inevitability of the imminent computability of the discrete algorithm, mainly focusing on the definition of a new theoretical model. The algorithm's core component is the multiplicative modular exponentiation with n bases and n exponents of which we discuss some cases.

*Email: cmme@fabbrini.org

2 Circular Multiplicative Modular Exponentiation (CMME)

2.1 The algorithm

Description Given the following openly accessible information:

1. a communication channel C
2. a prime p
3. a set of n primitive roots of p, $2 \leq n \leq \phi(p-1)$, $S_g = \{g_1, g_2, \ldots, g_n\}$
4. a direction of rotation D, which can be *clockwise* or *anticlockwise*[1] (*See sub-section* 2.4 for an explanation and example)

two parties denoted by A and B, using respectively the private $n-tuples$ $S_a = \{a_1, a_2, \ldots, a_n\}$, $S_b = \{b_1, b_2, \ldots, b_n\}$ and implenenting the public-domain CMME algorithm, are able to generate the two identical and secret $n-tuples$ $KA = \{KA_1, KA_2, \ldots, KA_n\}$, $KB = \{KB_1, KB_2, \ldots, KB_n\}$.

As stated in point 3, the lowest number of primitive roots employed is two. Starting with this instance, we show how the algorithm actually works before going on to the scenario where $n = 3$, and finally, generalizing to n bases.

Note: For the sake of clarity, we always omit *"mod p"*

2.2 Case n=2

Given a prime p, a couple of its primitive roots (g_1, g_2), A and B choose and keep secret two integers less than p, (a_1, a_2), (b_1, b_2), then:

1. A computes

$$A_1 = g_1^{a_1} g_2^{a_2}$$
$$A_2 = g_1^{a_2} g_2^{a_1}$$

2. B computes

$$B_1 = g_1^{b_1} g_2^{b_2}$$
$$B_2 = g_1^{b_2} g_2^{b_1}$$

3. A, B swap (A_1, A_2), (B_1, B_2), over the chosen public channel, then perform the same operation in $1, 2$ using the two swapped sets as bases

$$KA_1 = B_1^{a_1} B_2^{a_2}$$
$$KA_2 = B_1^{a_2} B_2^{a_1}$$

$$KB_1 = A_1^{b_1} A_2^{b_2}$$
$$KB_2 = A_1^{b_2} A_2^{b_1}$$

It's easy to verify that

$$KA_1 = KB_1$$
$$KA_2 = KB_2$$

Proof

$$KA_1 = B_1^{a_1} B_2^{a_2} = g_1^{b_1 a_1} g_2^{b_2 a_1} g_1^{b_2 a_2} g_2^{b_1 a_2} = g_1^{(b_1 a_1 + b_2 a_2)} g_2^{(b_2 a_1 + b_1 a_2)}$$
$$KB_1 = A_1^{b_1} A_2^{b_2} = g_1^{a_1 b_1} g_2^{a_2 b_1} g_1^{a_2 b_2} g_2^{a_1 b_2} = g_1^{(a_1 b_1 + a_2 b_2)} g_2^{(a_2 b_1 + a_1 b_2)}$$

$$\boxed{g_1^{(b_1 a_1 + b_2 a_2)} g_2^{(b_2 a_1 + b_1 a_2)} = g_1^{(a_1 b_1 + a_2 b_2)} g_2^{(a_2 b_1 + a_1 b_2)}}$$

$$KA_2 = B_1^{a_2} B_2^{a_1} = g_1^{b_1 a_2} g_2^{b_2 a_2} g_1^{b_2 a_1} g_2^{b_1 a_1} = g_1^{(b_1 a_2 + b_2 a_1)} g_2^{(b_2 a_2 + b_1 a_1)}$$
$$KB_2 = A_1^{b_2} A_2^{b_1} = g_1^{a_1 b_2} g_2^{a_2 b_2} g_1^{a_2 b_1} h_2^{a_1 b_1} = g_1^{(a_1 b_2 + a_2 b_1)} g_2^{(a_2 b_2 + a_1 b_1)}$$

$$\boxed{g_1^{(b_1 a_2 + b_2 a_1)} g_2^{(b_2 a_2 + b_1 a_1)} = g_1^{(a_1 b_2 + a_2 b_1)} g_2^{(a_2 b_2 + a_1 b_1)}}$$

Note: Obviously, only for this particular case, does it not make sense to choose a *"direction of rotation"*, but it is more correct to speak of a simple *"switch"* between the two positions.

[1] We only discuss the clockwise scenario in this essay.

2.3 Case n=3

Given a prime p, three of its primitive roots (g_1, g_2, g_3), A and B choose and keep secret three integers less than p, (a_1, a_2, a_3), (b_1, b_2, b_3), and agree on a "*clockwise*" direction of rotation (*see next sub-section*), then:

1. A computes

$$A_1 = g_1^{a_1} g_2^{a_2} g_3^{a_3}$$
$$A_2 = g_1^{a_3} g_2^{a_1} g_3^{a_2}$$
$$A_3 = g_1^{a_2} g_2^{a_3} g_3^{a_1}$$

2. B computes

$$B_1 = g_1^{b_1} g_2^{b_2} g_3^{b_3}$$
$$B_2 = g_1^{b_3} g_2^{b_1} g_3^{b_2}$$
$$B_3 = g_1^{b_2} g_2^{b_3} g_3^{b_1}$$

3. A, B swap (A_1, A_2, A_3), (B_1, B_2, B_3), then perform the same operation in $1, 2$ using the two swapped sets as bases

$$KA_1 = B_1^{a_1} B_2^{a_2} B_3^{a_3}$$
$$KA_2 = B_1^{a_3} B_2^{a_1} B_3^{a_2}$$
$$KA_3 = B_1^{a_2} B_2^{a_3} B_3^{a_1}$$

$$KB_1 = A_1^{b_1} A_2^{b_2} A_3^{b_3}$$
$$KB_2 = A_1^{b_3} A_2^{b_1} A_3^{b_2}$$
$$KB_3 = A_1^{b_2} A_2^{b_3} A_3^{b_1}$$

Now let's check that

$$KA_1 = KB_1$$
$$KA_2 = KB_2$$
$$KA_3 = KB_3$$

Proof

$$KA_1 = B_1^{a_1} B_2^{a_2} B_3^{a_3} = (g_1^{b_1 a_1} g_2^{b_2 a_1} g_3^{b_3 a_1})(g_1^{b_3 a_2} g_2^{b_1 a_2} g_3^{b_2 a_2})(g_1^{b_2 a_3} g_2^{b_3 a_3} g_3^{b_1 a_3}) =$$
$$= g_1^{(b_1 a_1 + b_3 a_2 + b_2 a_3)} g_2^{(b_2 a_1 + b_1 a_2 + b_3 a_3)} g_3^{(b_3 a_1 + b_2 a_2 + b_1 a_3)}$$

$$KB_1 = A_1^{b_1} A_2^{b_2} A_3^{b_3} = (g_1^{a_1 b_1} g_2^{a_2 b_1} g_3^{a_3 b_1})(g_1^{a_3 b_2} g_2^{a_1 b_2} g_3^{a_2 b_2})(g_1^{a_2 b_3} g_2^{a_3 b_3} g_3^{a_1 b_3}) =$$
$$= g_1^{(a_1 b_1 + a_3 b_2 + a_2 b_3)} g_2^{(a_2 b_1 + a_1 b_2 + a_3 b_3)} g_3^{(a_3 b_1 + a_2 b_2 + a_1 b_3)}$$

$$g_1^{(b_1 a_1 + b_3 a_2 + b_2 a_3)} g_2^{(b_2 a_1 + b_1 a_2 + b_3 a_3)} g_3^{(b_3 a_1 + b_2 a_2 + b_1 a_3)}$$
$$=$$
$$g_1^{(a_1 b_1 + a_3 b_2 + a_2 b_3)} g_2^{(a_2 b_1 + a_1 b_2 + a_3 b_3)} g_3^{(a_3 b_1 + a_2 b_2 + a_1 b_3)}$$

$$KA_2 = B_1^{a_3} B_2^{a_1} B_3^{a_2} = (g_1^{b_1 a_3} g_2^{b_2 a_3} g_3^{b_3 a_3})(g_1^{b_3 a_1} g_2^{b_1 a_1} g_3^{b_2 a_1})(g_1^{b_2 a_2} g_2^{b_3 a_2} g_3^{b_1 a_2}) =$$
$$= g_1^{(b_1 a_3 + b_3 a_1 + b_2 a_2)} g_2^{(b_2 a_3 + b_1 a_1 + b_3 a_2)} g_3^{(b_3 a_3 + b_2 a_1 + b_1 a_2)}$$

$$KB_2 = A_1^{b_3} A_2^{b_1} A_3^{b_2} = (g_1^{a_1 b_3} g_2^{a_2 b_3} g_3^{a_3 b_3})(g_1^{a_3 b_1} g_2^{a_1 b_1} g_3^{a_2 b_1})(g_1^{a_2 b_2} g_2^{a_3 b_2} g_3^{a_1 b_2}) =$$
$$= g_1^{(a_1 b_3 + a_3 b_1 + a_2 b_2)} g_2^{(a_2 b_3 + a_1 b_1 + a_3 b_2)} g_3^{(a_3 b_3 + a_2 b_1 + a_1 b_2)}$$

$$g_1^{(b_1 a_3 + b_3 a_1 + b_2 a_2)} g_2^{(b_2 a_3 + b_1 a_1 + b_3 a_2)} g_3^{(b_3 a_3 + b_2 a_1 + b_1 a_2)}$$
$$=$$
$$g_1^{(a_1 b_3 + a_3 b_1 + a_2 b_2)} g_2^{(a_2 b_3 + a_1 b_1 + a_3 b_2)} g_3^{(a_3 b_3 + a_2 b_1 + a_1 b_2)}$$

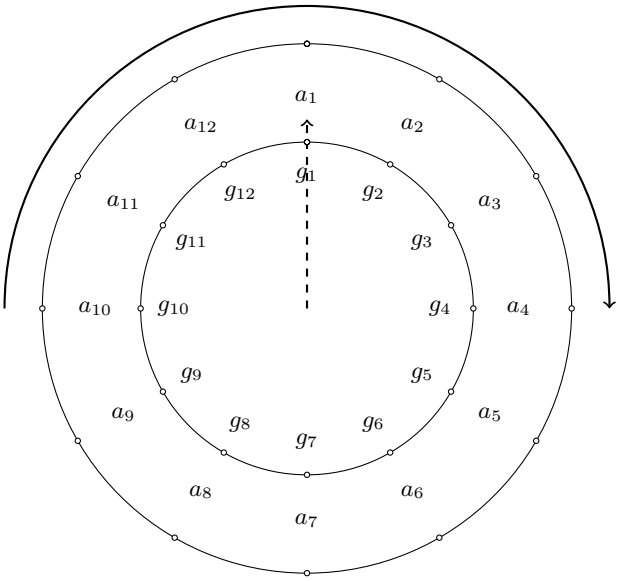
$$KA_3 = B_1^{a_2} B_2^{a_3} B_3^{a_1} = (g_1^{b_1 a_2} g_2^{b_2 a_2} g_3^{b_3 a_2})(g_1^{b_3 a_3} g_2^{b_1 a_3} g_3^{b_2 a_3})(g_1^{b_2 a_1} g_2^{b_3 a_1} g_3^{b_1 a_1}) =$$
$$= g_1^{(b_1 a_2 + b_3 a_3 + b_2 a_1)} g_2^{(b_2 a_2 + b_1 a_3 + b_3 a_1)} g_3^{(b_3 a_2 + b_2 a_3 + b_1 a_1)}$$

$$KB_3 = A_1^{b_2} A_2^{b_3} A_3^{b_1} = (g_1^{a_1 b_2} g_2^{a_2 b_2} g_3^{a_3 b_2})(g_1^{a_3 b_3} g_2^{a_1 b_3} g_3^{a_2 b_3})(g_1^{a_2 b_1} g_2^{a_3 b_1} g_3^{a_1 b_1}) =$$
$$= g_1^{(a_1 b_2 + a_3 b_3 + a_2 b_1)} g_2^{(a_2 b_2 + a_1 b_3 + a_3 b_1)} g_3^{(a_3 b_2 + a_2 b_3 + a_1 b_1)}$$

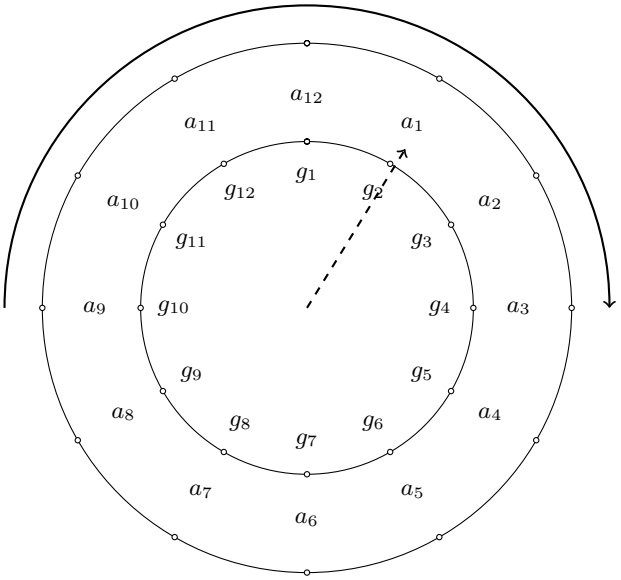
$$g_1^{(b_1 a_2 + b_3 a_3 + b_2 a_1)} g_2^{(b_2 a_2 + b_1 a_3 + b_3 a_1)} g_3^{(b_3 a_2 + b_2 a_3 + b_1 a_1)}$$
$$=$$
$$g_1^{(a_1 b_2 + a_3 b_3 + a_2 b_1)} g_2^{(a_2 b_2 + a_1 b_3 + a_3 b_1)} g_3^{(a_3 b_2 + a_2 b_3 + a_1 b_1)}$$

2.4 A graphical illustration of CMME's circularity

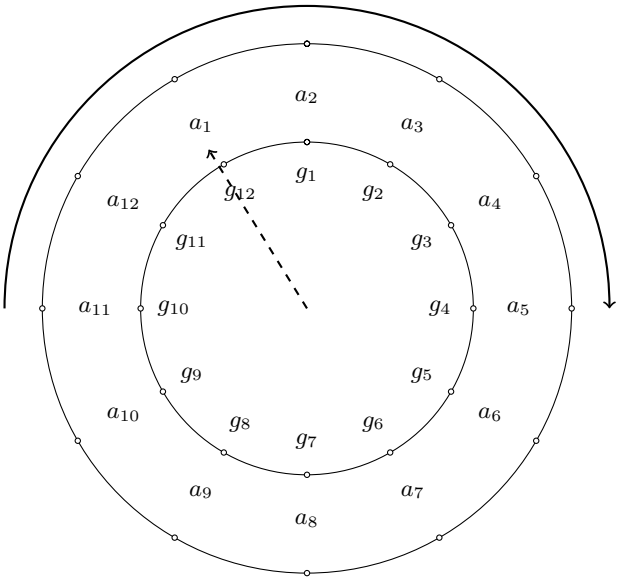
In order to illustrate the notion of circularity evoked by the algorithm's name, consider an example with twelve bases and relative exponents, in other words $n = 12$. Now let's examine the operations that A performs. Two concentric circles are drawn. The bases are placed on the inner circle, and the exponents on the outer one. The algorithm is currently in its first of 12 internal states. After performing the necessary calculations, the first element of the $12 - tuple$, that A will exchange with B, is generated.



In the next step, while the bases remain fixed, the exponents slide one position to the right, clockwise, A and B having agreed on this direction of rotation. At this point, similarly to what we just saw, the second element of the $A's$ public $12 - tuple$ is computed.



The procedure is repeated until the first exponent reaches the 12^{th} position and, after the last round of calculations, the $12 - tuple$ of A is complete and ready to be sent to B.



2.5 Generalization to n bases

Given a prime p, a $n-tuple$ of its primitive roots $(g_1, g_2, \ldots, g_n)$, A and B choose and keep secret n integers less than p, $(a_1, a_2, \ldots, a_n)$, $(b_1, b_2, \ldots, b_n)$, and agree on a "*clockwise*" direction of rotation, then:

1. A computes

$$\begin{gathered} A_1 = g_1^{a_1} \ldots g_n^{a_n} \\ A_2 = g_1^{a_n} \ldots g_n^{a_{n-1}} \\ \ldots \\ A_n = g_1^{a_2} \ldots g_n^{a_1} \end{gathered}$$

2. B computes

$$\begin{gathered} B_1 = g_1^{b_1} \ldots g_n^{b_n} \\ B_2 = g_1^{b_n} \ldots g_n^{b_{n-1}} \\ \ldots \\ B_n = g_1^{b_2} \ldots g_n^{b_1} \end{gathered}$$

3. A, B swap $(A_1, A_2, \ldots, A_n)$, $(B_1, B_2, \ldots, B_n)$, then perform the same operation in $1, 2$ using the two swapped sets as bases

$$\begin{gathered} KA_1 = B_1^{a_1} \ldots B_n^{a_n} \\ KA_2 = B_1^{a_n} \ldots B_n^{a_{n-1}} \\ \ldots \\ KA_n = B_1^{a_2} \ldots B_3^{a_1} \end{gathered}$$

$$\begin{gathered} KB_1 = A_1^{b_1} \ldots A_n^{b_n} \\ KB_2 = A_1^{b_n} \ldots A_n^{b_{n-1}} \\ \ldots \\ KB_n = A_1^{b_2} \ldots A_3^{b_1} \end{gathered}$$

By extending the results from the previous cases, we can expect

$$\begin{gathered} KA_1 = KB_1 \\ KA_2 = KB_2 \\ \ldots \\ KA_n = KB_n \end{gathered}$$

3 Security considerations

As already mentioned, the purpose of this paper is essentially to describe, at a theoretical level, an algorithm that is still in its infancy, which is what we did in the previous section. We consider CMME a useful addition to the research into new algorithm models for generating shared secret keys via insecure communication channels. Here, we only make a few observations while distinguishing between the present and the near future, leaving a more thorough examination of its security to cryptanalysis.

3.1 Pre-quantum

The current era might be regarded as a "pre-quantum" one, in the sense that the development of quantum computing, while determined in its fundamental principles, lacks the necessary hardware to fully realize its immense potential. As illustrated so far, CMME can also be seen as a projection of the Diffie-Hellman exchange from a one-dimensional space to a multidimensional one. In fact, while DHKE provides for a single generator, in CMME the minimum number of generators is two. Consequently, while in the former the key is an integer, in the latter it is an $n-dimensional$ vector. After establishing the link between the two schemes, we can state that CMME can offer at least the same level of security as the DH exchange due to the fact that the modular exponentiation also serves as the foundation for CMME's design. Likewise, the same recommendations made for the Diffie-Hellman model regarding the selection of the prime number p apply to CMME as well. [11] [12]

3.2 Post-quantum

We are now referring to a near future when, thanks to Shor's algorithm, quantum computers equipped with a sufficient number of *Qubits* will be able to easily calculate discrete logarithms. Firstly, it should be noted that by multiplying the public keys of one of the two parties, say A, and being able to compute discrete logarithms, it is simple to obtain the sum of A's secret exponents using the product of the public primitive roots as base. On the other hand, getting the single addends is not so easy. Let us take for example the case $n = 2$

$$A_1 = g_1^{a_1} g_2^{a_2}$$
$$A_2 = g_1^{a_2} g_2^{a_1}$$

$$A_1 A_2 = g_1^{a_1} g_2^{a_2} g_1^{a_2} g_2^{a_1} = g_1^{(a_1+a_2)} g_2^{(a_2+a_1)} = (g_1 g_2)^{(a_1+a_2)}$$

Given the structure of CMME, in a post-quantum era, we believe that the search for the secret keys of A and B requires an additional effort compared to models based on the classic Diffie-Hellman exchange.

4 Conclusions

We have described a theoretical model of an algorithm capable of generating a shared secret key between A and B with only public channels of communication available. In addition, we have shown how CMME and the Diffie-Hellman model relate one another and highlighted how, given a prime number, at least two of its public primitive roots, an equal number of private exponents and a direction of rotation, CMME can achieve commutativity through the multiplication of the modular exponentiation, graphically demonstrating the circularity implied by the algorithm's name.

References

[1] Merkle, Ralph C. *244 Project Proposal*, UC Berkeley, Fall 1974

[2] Diffie, Whitfield, and Martin E. Hellman *New directions in cryptography*, IEEE Trans. Inform. Theory, vol. IT-22, pp. 644-654, Nov. 1976.

[3] Ellis, James H. *The possibility of secure non-secret digital encryption* UK Communications Electronics Security Group 8 1970.

[4] Robert Hannigan *Front doors and strong locks: encryption, privacy and intelligence gathering in the digital era* MIT speech, March 2016.

[5] Partala, Juha *Algebraic generalization of Diffie–Hellman key exchange* Journal of Mathematical Cryptology 12.1 (2018): 1-21

[6] Coppersmith, Don, Andrew M. Odlzyko, and Richard Schroeppel *Discrete logarithms in GF (p)* Algorithmica 1.1-4 (1986): 1-15.

[7] Koblitz, Neal *Elliptic curve cryptosystems*, Mathematics of computation 48.177 (1987): 203-209.

[8] Miller, V. S. *Use of elliptic curves in cryptography*, Advances in Cryptography CRYPTO'85 (Lecture Notes in Computer Science, vol 218)." (1986): 417-426.

[9] Jao, David, and De Feo, Luca *Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies*, Post-Quantum Cryptography: 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29–December 2, 2011. Proceedings 4. Springer Berlin Heidelberg, 2011.

[10] Shor, Peter W. *Algorithms for quantum computation: discrete logarithms and factoring*, Proceedings 35th annual symposium on foundations of computer science. Ieee, 1994.

[11] Den Boer, Bert *Diffie-Hellman is as strong as discrete log for certain primes*, Advances in Cryptology—CRYPTO'88: Proceedings 8. Springer New York, 1990.

[12] Adrian, David, et al. *Imperfect forward secrecy: How Diffie-Hellman fails in practice*, Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. 2015.